Investigate_a_Dataset

January 13, 2019

1 Project: Investigate a Dataset (Replace this with something more specific!)

1.1 Table of Contents

Introduction
Data Wrangling
Exploratory Data Analysis
Conclusions
Introduction - TMDb movie data (cleaned from original data on Kaggle)
This data set contains information about 10,000 movies collected from The Movie Database (TMDb), including user ratings and revenue
Questions: How did the release of films change over time? What kinds of properties are associated with movies that have high investments?

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Data Wrangling

> **Tip**: In this section of the report, you will load in the data, check for cleanliness, and then trim and clean your dataset for analysis. Make sure that you document your steps carefully and justify your cleaning decisions.

1.1.1 General Properties

```
In [2]: # Loading data
        df = pd.read_csv('tmdb-movies.csv')
```

```
In [3]: df.head()
        # For the questions, we have to split the column genres
```

```
Out[3]:        id    imdb_id  popularity     budget     revenue  \
        0  135397  tt0369610   32.985763  150000000  1513528810
        1   76341  tt1392190   28.419936  150000000   378436354
```

```
2  262500  tt2908446   13.112507  110000000   295238201
3  140607  tt2488496   11.173104  200000000  2068178225
4  168259  tt2820852    9.335014  190000000  1506249360

                original_title  \
0                Jurassic World
1            Mad Max: Fury Road
2                      Insurgent
3  Star Wars: The Force Awakens
4                      Furious 7

                                                 cast  \
0  Chris Pratt|Bryce Dallas Howard|Irrfan Khan|Vi...
1  Tom Hardy|Charlize Theron|Hugh Keays-Byrne|Nic...
2  Shailene Woodley|Theo James|Kate Winslet|Ansel...
3  Harrison Ford|Mark Hamill|Carrie Fisher|Adam D...
4  Vin Diesel|Paul Walker|Jason Statham|Michelle ...

                                       homepage          director  \
0                    http://www.jurassicworld.com/   Colin Trevorrow
1                      http://www.madmaxmovie.com/     George Miller
2    http://www.thedivergentseries.movie/#insurgent  Robert Schwentke
3  http://www.starwars.com/films/star-wars-episod...       J.J. Abrams
4                          http://www.furious7.com/         James Wan

                        tagline      ...       \
0              The park is open.      ...
1             What a Lovely Day.      ...
2    One Choice Can Destroy You      ...
3  Every generation has a story.      ...
4            Vengeance Hits Home      ...

                                          overview runtime  \
0  Twenty-two years after the events of Jurassic ...     124
1  An apocalyptic story set in the furthest reach...     120
2  Beatrice Prior must confront her inner demons ...     119
3  Thirty years after defeating the Galactic Empi...     136
4  Deckard Shaw seeks revenge against Dominic Tor...     137

                                     genres  \
0  Action|Adventure|Science Fiction|Thriller
1  Action|Adventure|Science Fiction|Thriller
2         Adventure|Science Fiction|Thriller
3   Action|Adventure|Science Fiction|Fantasy
4                     Action|Crime|Thriller

                              production_companies release_date vote_count  \
0  Universal Studios|Amblin Entertainment|Legenda...       6/9/15       5562
```

```
1  Village Roadshow Pictures|Kennedy Miller Produ...      5/13/15       6185
2  Summit Entertainment|Mandeville Films|Red Wago...      3/18/15       2480
3         Lucasfilm|Truenorth Productions|Bad Robot     12/15/15       5292
4  Universal Pictures|Original Film|Media Rights ...       4/1/15       2947

   vote_average  release_year    budget_adj   revenue_adj
0           6.5          2015  1.379999e+08  1.392446e+09
1           7.1          2015  1.379999e+08  3.481613e+08
2           6.3          2015  1.012000e+08  2.716190e+08
3           7.5          2015  1.839999e+08  1.902723e+09
4           7.3          2015  1.747999e+08  1.385749e+09

[5 rows x 21 columns]
```

```
In [4]: df.shape
        # In the dataset, there are 10866 rows and 21 columns

Out[4]: (10866, 21)

In [5]: df.info()
        # there are some missing values

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
Data columns (total 21 columns):
id                      10866 non-null int64
imdb_id                 10856 non-null object
popularity              10866 non-null float64
budget                  10866 non-null int64
revenue                 10866 non-null int64
original_title          10866 non-null object
cast                    10790 non-null object
homepage                2936 non-null object
director                10822 non-null object
tagline                 8042 non-null object
keywords                9373 non-null object
overview                10862 non-null object
runtime                 10866 non-null int64
genres                  10843 non-null object
production_companies    9836 non-null object
release_date            10866 non-null object
vote_count              10866 non-null int64
vote_average            10866 non-null float64
release_year            10866 non-null int64
budget_adj              10866 non-null float64
revenue_adj             10866 non-null float64
dtypes: float64(4), int64(6), object(11)
memory usage: 1.7+ MB
```

```
In [6]: sum(df.duplicated())
        # one duplicate is found

Out[6]: 1

In [7]: df.describe()
        # some of these infomations are unusable like the coulumn release_year

Out[7]:                   id    popularity        budget       revenue       runtime  \
        count   10866.000000  10866.000000  1.086600e+04  1.086600e+04  10866.000000
        mean    66064.177434      0.646441  1.462570e+07  3.982332e+07    102.070863
        std     92130.136561      1.000185  3.091321e+07  1.170035e+08     31.381405
        min         5.000000      0.000065  0.000000e+00  0.000000e+00      0.000000
        25%     10596.250000      0.207583  0.000000e+00  0.000000e+00     90.000000
        50%     20669.000000      0.383856  0.000000e+00  0.000000e+00     99.000000
        75%     75610.000000      0.713817  1.500000e+07  2.400000e+07    111.000000
        max    417859.000000     32.985763  4.250000e+08  2.781506e+09    900.000000

                 vote_count  vote_average  release_year    budget_adj   revenue_adj
        count  10866.000000  10866.000000  10866.000000  1.086600e+04  1.086600e+04
        mean     217.389748      5.974922   2001.322658  1.755104e+07  5.136436e+07
        std      575.619058      0.935142     12.812941  3.430616e+07  1.446325e+08
        min       10.000000      1.500000   1960.000000  0.000000e+00  0.000000e+00
        25%       17.000000      5.400000   1995.000000  0.000000e+00  0.000000e+00
        50%       38.000000      6.000000   2006.000000  0.000000e+00  0.000000e+00
        75%      145.750000      6.600000   2011.000000  2.085325e+07  3.369710e+07
        max     9767.000000      9.200000   2015.000000  4.250000e+08  2.827124e+09
```

1.1.2 Data Cleaning

- split the coulumn genre
- drop the columns: id, imdb_id, cast, homepage, director, tagline, overview, production_companies, release_date, vote_count, vote_average, budget_adj, revenue_adj
- fill in missing values
- drop duplicates

```
In [8]: # drop columns
        df.drop(['id', 'imdb_id', 'cast', 'homepage', 'director', 'tagline', 'overview', 'produc

In [ ]:

In [9]: # use means to fill in missing vaues
        df.fillna(df.mean(),inplace=True)

Out[9]:    popularity     budget     revenue  \
        0    32.985763  150000000  1513528810
        1    28.419936  150000000   378436354
        2    13.112507  110000000   295238201
        3    11.173104  200000000  2068178225
```

4	9.335014	190000000	1506249360
5	9.110700	135000000	532950503
6	8.654359	155000000	440603537
7	7.667400	108000000	595380321
8	7.404165	74000000	1156730962
9	6.326804	175000000	853708609
10	6.200282	245000000	880674609
11	6.189369	176000003	183987723
12	6.118847	15000000	36869414
13	5.984995	88000000	243637091
14	5.944927	280000000	1405035767
15	5.898400	44000000	155760117
16	5.749758	48000000	325771424
17	5.573184	130000000	518602163
18	5.556818	95000000	542351353
19	5.476958	160000000	650523427
20	5.462138	190000000	209035668
21	5.337064	30000000	91709827
22	4.907832	110000000	470490832
23	4.710402	40000000	569651467
24	4.648046	28000000	133346506
25	4.566713	150000000	682330139
26	4.564549	68000000	215863606
27	4.503789	81000000	403802136
28	4.062293	20000000	88346473
29	3.968891	61000000	311256926
...	...	...	...
10836	0.239435	0	0
10837	0.291704	0	0
10838	0.151845	0	0
10839	0.276133	0	0
10840	0.102530	0	0
10841	0.264925	75000	0
10842	0.253437	0	0
10843	0.252399	0	0
10844	0.236098	0	0
10845	0.230873	0	0
10846	0.212716	0	0
10847	0.034555	0	0
10848	0.207257	5115000	12000000
10849	0.206537	0	0
10850	0.202473	0	0
10851	0.342791	0	0
10852	0.227220	0	0
10853	0.163592	0	0
10854	0.146402	0	0
10855	0.141026	700000	0
10856	0.140934	0	0

```
10857    0.131378          0           0
10858    0.317824          0           0
10859    0.089072          0           0
10860    0.087034          0           0
10861    0.080598          0           0
10862    0.065543          0           0
10863    0.065141          0           0
10864    0.064317          0           0
10865    0.035919      19000           0

                                      original_title  runtime  \
0                                      Jurassic World      124
1                                  Mad Max: Fury Road      120
2                                           Insurgent      119
3                        Star Wars: The Force Awakens      136
4                                           Furious 7      137
5                                       The Revenant      156
6                                  Terminator Genisys      125
7                                         The Martian      141
8                                             Minions       91
9                                           Inside Out       94
10                                             Spectre      148
11                                    Jupiter Ascending      124
12                                           Ex Machina      108
13                                               Pixels      105
14                           Avengers: Age of Ultron      141
15                                 The Hateful Eight      167
16                                          Taken 3      109
17                                          Ant-Man      115
18                                       Cinderella      112
19      The Hunger Games: Mockingjay - Part 2      136
20                                     Tomorrowland      130
21                                         Southpaw      123
22                                       San Andreas      114
23                          Fifty Shades of Grey      125
24                                   The Big Short      130
25         Mission: Impossible - Rogue Nation      131
26                                           Ted 2      115
27                 Kingsman: The Secret Service      130
28                                       Spotlight      128
29              Maze Runner: The Scorch Trials      132
...                                           ...      ...
10836                             Walk Don't Run      114
10837                               The Blue Max      156
10838                           The Professionals      117
10839  It's the Great Pumpkin, Charlie Brown       25
10840                           Funeral in Berlin      102
10841                                The Shooting       82
```

10842	Winnie the Pooh and the Honey Tree	25
10843	Khartoum	134
10844	Our Man Flint	108
10845	Carry On Cowboy	93
10846	Dracula: Prince of Darkness	90
10847	Island of Terror	89
10848	Fantastic Voyage	100
10849	Gambit	109
10850	Harper	121
10851	Born Free	95
10852	A Big Hand for the Little Lady	95
10853	Alfie	114
10854	The Chase	135
10855	The Ghost & Mr. Chicken	90
10856	The Ugly Dachshund	93
10857	Nevada Smith	128
10858	The Russians Are Coming, The Russians Are Coming	126
10859	Seconds	100
10860	Carry On Screaming!	87
10861	The Endless Summer	95
10862	Grand Prix	176
10863	Beregis Avtomobilya	94
10864	What's Up, Tiger Lily?	80
10865	Manos: The Hands of Fate	74

	genres	release_year
0	Action|Adventure|Science Fiction|Thriller	2015
1	Action|Adventure|Science Fiction|Thriller	2015
2	Adventure|Science Fiction|Thriller	2015
3	Action|Adventure|Science Fiction|Fantasy	2015
4	Action|Crime|Thriller	2015
5	Western|Drama|Adventure|Thriller	2015
6	Science Fiction|Action|Thriller|Adventure	2015
7	Drama|Adventure|Science Fiction	2015
8	Family|Animation|Adventure|Comedy	2015
9	Comedy|Animation|Family	2015
10	Action|Adventure|Crime	2015
11	Science Fiction|Fantasy|Action|Adventure	2015
12	Drama|Science Fiction	2015
13	Action|Comedy|Science Fiction	2015
14	Action|Adventure|Science Fiction	2015
15	Crime|Drama|Mystery|Western	2015
16	Crime|Action|Thriller	2015
17	Science Fiction|Action|Adventure	2015
18	Romance|Fantasy|Family|Drama	2015
19	War|Adventure|Science Fiction	2015
20	Action|Family|Science Fiction|Adventure|Mystery	2015
21	Action|Drama	2015

```
22                                Action|Drama|Thriller  2015
23                                       Drama|Romance  2015
24                                        Comedy|Drama  2015
25                                              Action  2015
26                                              Comedy  2015
27                         Crime|Comedy|Action|Adventure  2015
28                                Drama|Thriller|History  2015
29                       Action|Science Fiction|Thriller  2015
...                                                 ...   ...
10836                                    Comedy|Romance  1966
10837                         War|Action|Adventure|Drama  1966
10838                           Action|Adventure|Western  1966
10839                                   Family|Animation  1966
10840                                           Thriller  1966
10841                                            Western  1966
10842                                   Animation|Family  1966
10843               Adventure|Drama|War|History|Action  1966
10844          Adventure|Comedy|Fantasy|Science Fiction  1966
10845                                    Comedy|Western  1966
10846                                            Horror  1966
10847                           Science Fiction|Horror  1966
10848                         Adventure|Science Fiction  1966
10849                               Action|Comedy|Crime  1966
10850              Action|Drama|Thriller|Crime|Mystery  1966
10851             Adventure|Drama|Action|Family|Foreign  1966
10852                                            Western  1966
10853                               Comedy|Drama|Romance  1966
10854                               Thriller|Drama|Crime  1966
10855                     Comedy|Family|Mystery|Romance  1966
10856                               Comedy|Drama|Family  1966
10857                                    Action|Western  1966
10858                                       Comedy|War  1966
10859            Mystery|Science Fiction|Thriller|Drama  1966
10860                                           Comedy  1966
10861                                      Documentary  1966
10862                           Action|Adventure|Drama  1966
10863                                   Mystery|Comedy  1966
10864                                    Action|Comedy  1966
10865                                           Horror  1966

[10866 rows x 7 columns]
```

```
In [10]: # check missing values
         df.info()
         # only in genres there are also missing values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10866 entries, 0 to 10865
```

```
Data columns (total 7 columns):
popularity        10866 non-null float64
budget            10866 non-null int64
revenue           10866 non-null int64
original_title    10866 non-null object
runtime           10866 non-null int64
genres            10843 non-null object
release_year      10866 non-null int64
dtypes: float64(1), int64(4), object(2)
memory usage: 594.3+ KB
```

```
In [11]: df[df.genres.isnull()]
```

Out[11]:

	popularity	budget	revenue	original_title	runtime	genres
424	0.244648	0	0	Belli di papÃ	100	NaN
620	0.129696	0	0	All Hallows' Eve 2	90	NaN
997	0.330431	0	0	Star Wars Rebels: Spark of Rebellion	44	NaN
1712	0.302095	0	0	Prayers for Bobby	88	NaN
1897	0.020701	0	0	Jonas Brothers: The Concert Experience	76	NaN
2370	0.081892	0	0	Freshman Father	0	NaN
2376	0.068411	0	0	Doctor Who: A Christmas Carol	62	NaN
2853	0.130018	0	0	Vizontele	110	NaN
3279	0.145331	0	0			
4547	0.520520	0	0			
4732	0.235911	0	0			
4797	0.167501	0	0			
4890	0.083202	0	0			
5830	0.248944	0	0			
5934	0.067433	0	0			
6043	0.039080	0	0			
6530	0.092724	0	0			
8234	0.028874	0	0			
8614	0.273934	0	0			
8878	0.038045	0	0			
9307	0.094652	0	0			
9799	0.175008	0	0			
10659	0.344172	5000	0			

```
3279                                        ìêÿřì ë       96    NaN
4547   London 2012 Olympic Opening Ceremony: Isles of...      220    NaN
4732                                    The Scapegoat      100    NaN
4797                            Doctor Who: The Snowmen       60    NaN
4890                          Cousin Ben Troop Screening       2    NaN
5830                Doctor Who: The Time of the Doctor       60    NaN
5934                                    Prada: Candy         3    NaN
6043                                  Bombay Talkies      127    NaN
6530                                    Saw Rebirth         6    NaN
8234                                  Viaggi di nozze      103    NaN
8614                         T2 3-D: Battle Across Time       12    NaN
8878                   Mom's Got a Date With a Vampire       85    NaN
9307                                       Goldeneye      105    NaN
9799                                    The Amputee         5    NaN
10659                     The Party at Kitty and Stud's       71    NaN

       release_year
424            2015
620            2015
997            2014
1712           2009
1897           2009
2370           2010
2376           2010
2853           2001
3279           2008
4547           2012
4732           2012
4797           2012
4890           2012
5830           2013
5934           2013
6043           2013
6530           2005
8234           1995
8614           1996
8878           2000
9307           1989
9799           1974
10659          1970
```

In [12]:
```
# drop rows with missing values in the column genre
df.dropna(inplace=True)
# check missing values
df.info()
#no more missing values
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 10843 entries, 0 to 10865
```

```
Data columns (total 7 columns):
popularity        10843 non-null float64
budget            10843 non-null int64
revenue           10843 non-null int64
original_title    10843 non-null object
runtime           10843 non-null int64
genres            10843 non-null object
release_year      10843 non-null int64
dtypes: float64(1), int64(4), object(2)
memory usage: 677.7+ KB
```

```
In [13]: # drop the duplicate
         df.drop_duplicates(inplace=True)

In [14]: sum(df.duplicated())
         # no more duplicates in the dataset

Out[14]: 0

In [15]: # splitting the column genres
         # splitting the column genres
         # drop the old column genres
         #transform numeric coulmns into separate rows and "copy" the other columns
             #drop the column variable
             # drop the empty values
         df.split = df['genres'].str.split('|', expand=True) \
             .merge(df, left_index = True, right_index = True) \
             .drop(["genres"], axis = 1) \
             .melt(id_vars = ['popularity', 'budget', 'revenue', 'original_title', 'runtime', 'r
             .drop(["variable"], axis = 1) \
             .dropna()


         # now there are all the genres in seperate rows



In [16]: df.split

Out[16]:        popularity     budget     revenue  \
         0       32.985763  150000000  1513528810
         1       28.419936  150000000   378436354
         2       13.112507  110000000   295238201
         3       11.173104  200000000  2068178225
         4        9.335014  190000000  1506249360
         5        9.110700  135000000   532950503
         6        8.654359  155000000   440603537
         7        7.667400  108000000   595380321
```

8	7.404165	74000000	1156730962
9	6.326804	175000000	853708609
10	6.200282	245000000	880674609
11	6.189369	176000003	183987723
12	6.118847	15000000	36869414
13	5.984995	88000000	243637091
14	5.944927	280000000	1405035767
15	5.898400	44000000	155760117
16	5.749758	48000000	325771424
17	5.573184	130000000	518602163
18	5.556818	95000000	542351353
19	5.476958	160000000	650523427
20	5.462138	190000000	209035668
21	5.337064	30000000	91709827
22	4.907832	110000000	470490832
23	4.710402	40000000	569651467
24	4.648046	28000000	133346506
25	4.566713	150000000	682330139
26	4.564549	68000000	215863606
27	4.503789	81000000	403802136
28	4.062293	20000000	88346473
29	3.968891	61000000	311256926
...	...	...	...
53722	0.151355	30	0
53727	0.271575	0	0
53756	0.438314	0	0
53763	0.241283	18000000	9000000
53768	0.337261	0	0
53791	0.664326	4000000	5000000
53795	0.628520	0	0
53840	0.772494	113	115103979
53884	0.347800	2700000	3500000
53908	0.459144	0	0
53930	0.058402	0	0
53943	0.971417	0	0
53962	0.383880	0	11000000
53996	0.492877	0	5200000
54001	0.381352	25485000	29548291
54027	0.429246	10000000	1500000
54048	0.243896	5760000	0
54067	0.004770	0	0
54078	0.411816	0	0
54087	0.267140	0	0
54091	0.148940	0	0
54092	0.141056	0	0
54105	1.090065	0	49579269
54112	0.573763	0	0
54132	0.277769	24000000	21049053

54171	0.410366	1377800	0
54179	0.299911	12000000	20000000
54187	0.252399	0	0
54194	0.202473	0	0
54195	0.342791	0	0

	original_title	runtime	release_year	\
0	Jurassic World	124	2015	
1	Mad Max: Fury Road	120	2015	
2	Insurgent	119	2015	
3	Star Wars: The Force Awakens	136	2015	
4	Furious 7	137	2015	
5	The Revenant	156	2015	
6	Terminator Genisys	125	2015	
7	The Martian	141	2015	
8	Minions	91	2015	
9	Inside Out	94	2015	
10	Spectre	148	2015	
11	Jupiter Ascending	124	2015	
12	Ex Machina	108	2015	
13	Pixels	105	2015	
14	Avengers: Age of Ultron	141	2015	
15	The Hateful Eight	167	2015	
16	Taken 3	109	2015	
17	Ant-Man	115	2015	
18	Cinderella	112	2015	
19	The Hunger Games: Mockingjay - Part 2	136	2015	
20	Tomorrowland	130	2015	
21	Southpaw	123	2015	
22	San Andreas	114	2015	
23	Fifty Shades of Grey	125	2015	
24	The Big Short	130	2015	
25	Mission: Impossible - Rogue Nation	131	2015	
26	Ted 2	115	2015	
27	Kingsman: The Secret Service	130	2015	
28	Spotlight	128	2015	
29	Maze Runner: The Scorch Trials	132	2015	
...	...	...	...	
53722	The Meteor Man	100	1993	
53727	Boiling Point	92	1993	
53756	The Night of the Generals	148	1967	
53763	Doctor Dolittle	152	1967	
53768	Point Blank	92	1967	
53791	The Great Escape	172	1963	
53795	Murder at the Gallop	81	1963	
53840	The Karate Kid, Part II	113	1986	
53884	The Wraith	93	1986	
53908	Legal Eagles	116	1986	

53930	Dead End Drive-In	87	1986
53943	Westworld	88	1973
53962	The Golden Voyage of Sinbad	105	1973
53996	Kelly's Heroes	144	1970
54001	Tora! Tora! Tora!	144	1970
54027	The Private Life of Sherlock Holmes	125	1970
54048	Von Ryan's Express	117	1965
54067	Die, Monster, Die!	80	1965
54078	Paint Your Wagon	158	1969
54087	The Valley of Gwangi	96	1969
54091	Castle Keep	105	1969
54092	Hercules in New York	91	1969
54105	Revenge of the Pink Panther	104	1978
54112	The Star Wars Holiday Special	97	1978
54132	The Wiz	134	1978
54171	Batman	105	1966
54179	The Sand Pebbles	182	1966
54187	Khartoum	134	1966
54194	Harper	121	1966
54195	Born Free	95	1966

	genres
0	Action
1	Action
2	Adventure
3	Action
4	Action
5	Western
6	Science Fiction
7	Drama
8	Family
9	Comedy
10	Action
11	Science Fiction
12	Drama
13	Action
14	Action
15	Crime
16	Crime
17	Science Fiction
18	Romance
19	War
20	Action
21	Action
22	Action
23	Drama
24	Comedy
25	Action

```
26              Comedy
27               Crime
28               Drama
29              Action
...                ...
53722           Family
53727         Thriller
53756          Mystery
53763            Music
53768          Mystery
53791              War
53795           Comedy
53840           Family
53884            Crime
53908         Thriller
53930         Thriller
53943  Science Fiction
53962          Fantasy
53996              War
54001              War
54027          Romance
54048              War
54067          Foreign
54078            Music
54087          Western
54091              War
54092  Science Fiction
54105           Family
54112         TV Movie
54132  Science Fiction
54171            Crime
54179          Romance
54187           Action
54194          Mystery
54195          Foreign

[26955 rows x 7 columns]
```

In [17]:
```
#check
df.split.query('original_title == "Jurassic World"')
```

Out[17]:

	popularity	budget	revenue	original_title	runtime	release_year	genres
0	32.985763	150000000	1513528810	Jurassic World	124		
10842	32.985763	150000000	1513528810	Jurassic World	124		
21684	32.985763	150000000	1513528810	Jurassic World	124		
32526	32.985763	150000000	1513528810	Jurassic World	124		

```
0              2015           Action
10842          2015        Adventure
21684          2015  Science Fiction
32526          2015         Thriller
```

In [ ]:

Exploratory Data Analysis

Tip: Now that you've trimmed and cleaned your data, you're ready to move on to exploration. Compute statistics and create visualizations with the goal of addressing the research questions that you posed in the Introduction section. It is recommended that you be systematic with your approach. Look at one variable at a time, and then follow it up by looking at relationships between variables.

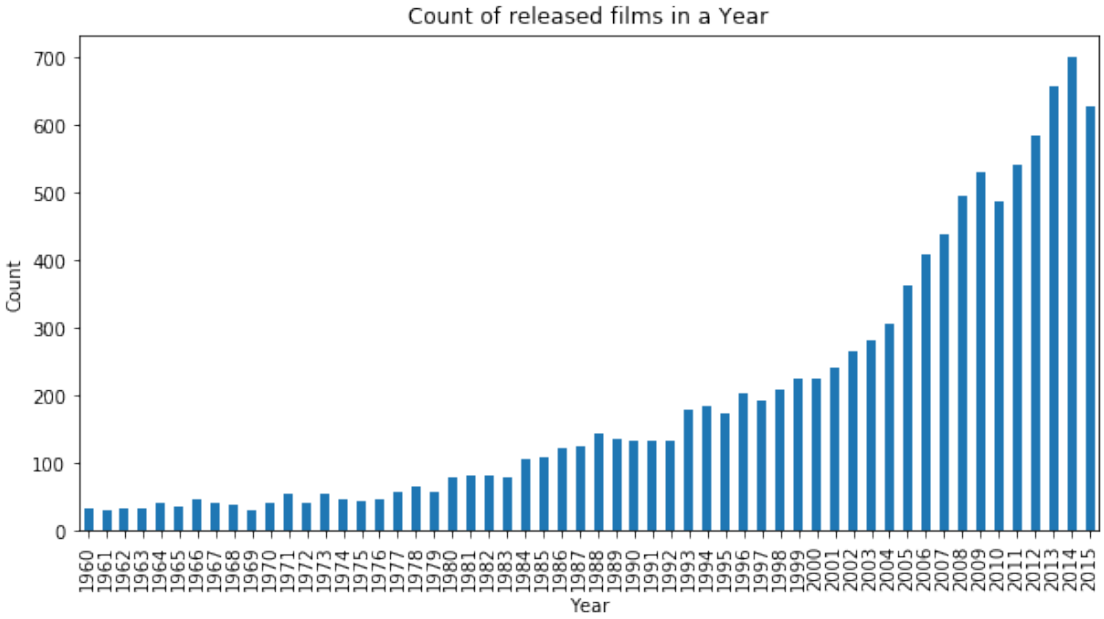
1.1.3 Research Question 1

How did the release of films change over time?

In [18]:
```
# Question: How many movies were released per year?
```

In [19]:
```
released_films = df.groupby(['release_year']).count()['genres'].plot(kind='bar', title
released_films.set_xlabel("Year")
released_films.set_ylabel("Count")
```

Out[19]: Text(0,0.5,'Count')



In [20]:
```
# Answer: Over time, more and more films were released in one year. Most films were rel
```

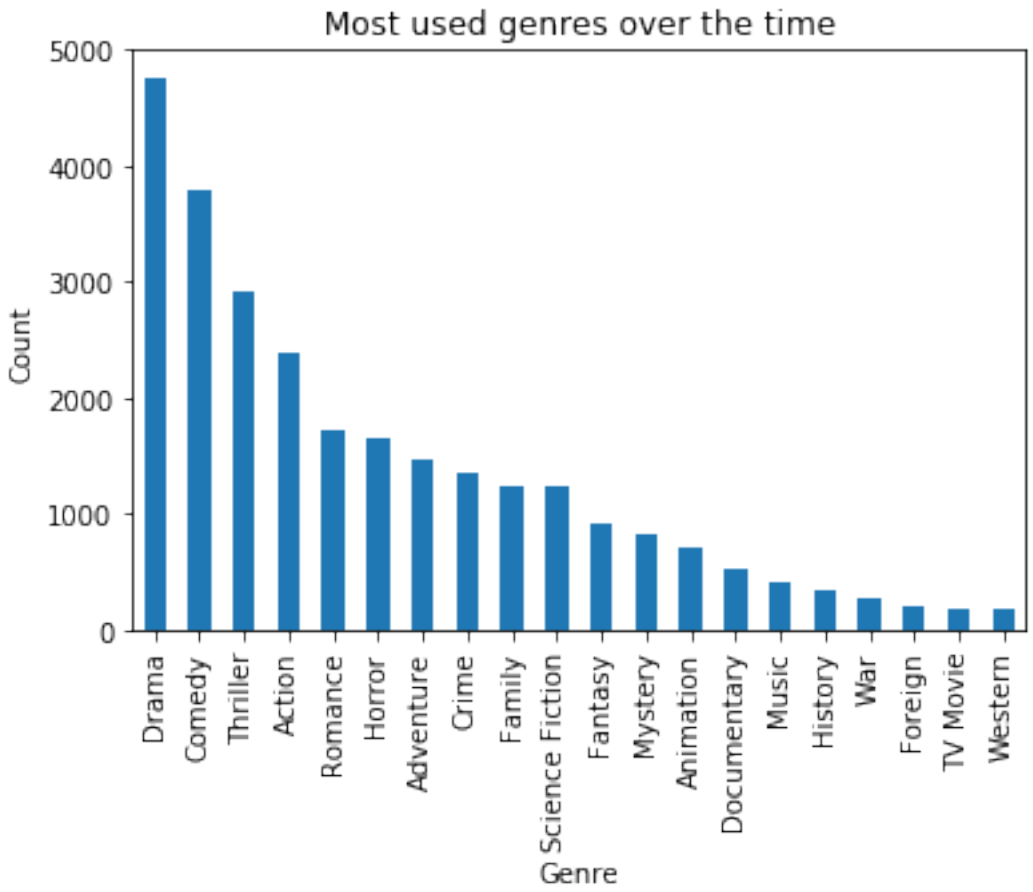
In [21]:
```
#Question: Which genre is the most used for films over the time?
# Note: Films with a single genre as well as films with several genres are considered.
genre_used = df.split.genres.value_counts()
genre_used
```

Out[21]:
```
Drama              4760
Comedy             3793
Thriller           2907
Action             2384
Romance            1712
Horror             1637
Adventure          1471
Crime              1354
Family             1231
Science Fiction    1229
Fantasy             916
Mystery             810
Animation           699
Documentary         520
Music               408
History             334
War                 270
Foreign             188
TV Movie            167
Western             165
Name: genres, dtype: int64
```

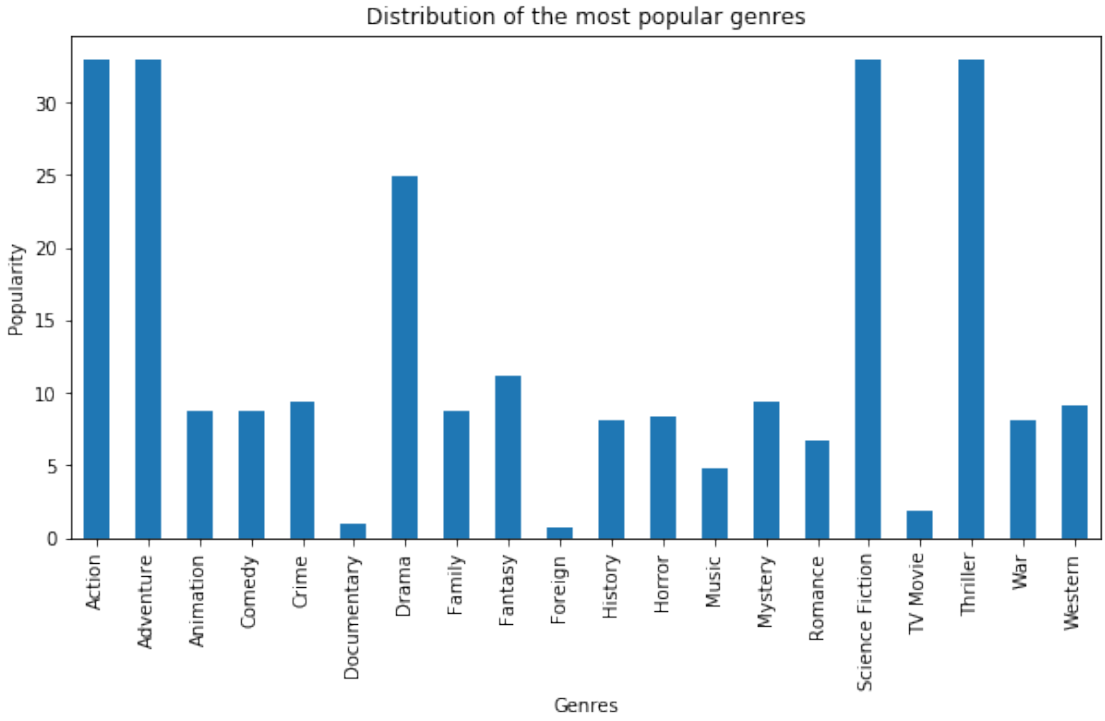
In [22]:
```
# Answer: Most of the films over the period belong to the drama genre.
```

In [23]:
```
# Visualization of question 1
genre_used = df.split.genres.value_counts().plot(kind='bar', title ="Most used genres o
genre_used.set_xlabel("Genre")
genre_used.set_ylabel("Count")
```

Out[23]:
```
Text(0,0.5,'Count')
```



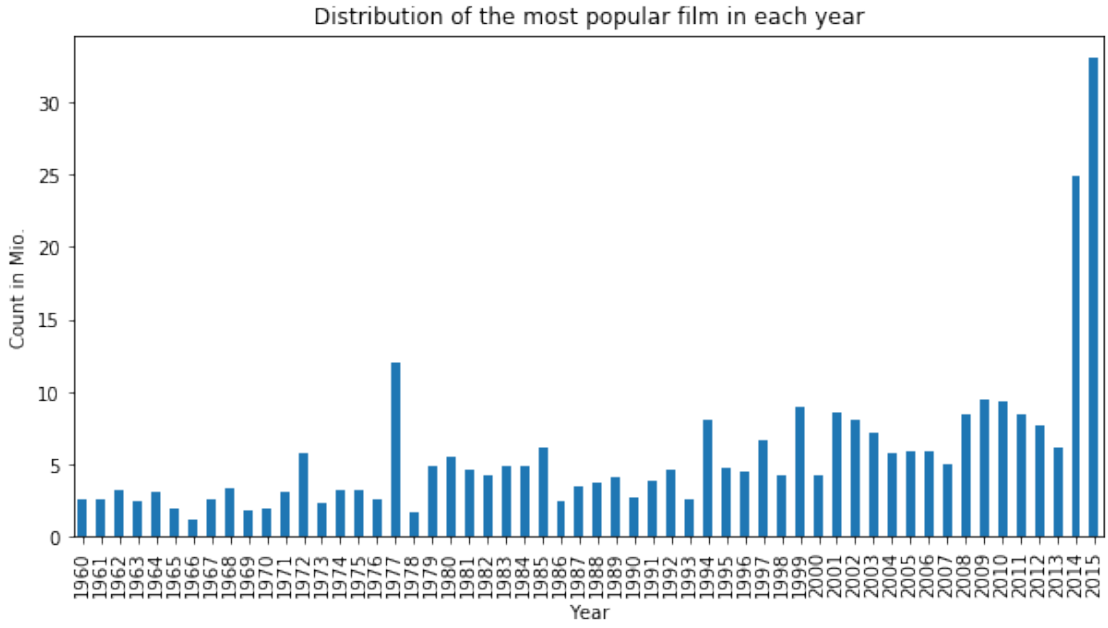

```
In [24]: # The Visualization shows, that most of the films over the period belog tpo the drama g
         # The least used genre is Western, with only 165 films.

In [25]: # Most popular genre over the time

In [26]: popularity_genre = df.split.groupby(['genres']).max().popularity.plot(kind='bar', title
         popularity_genre.set_xlabel("Genres")
         popularity_genre.set_ylabel("Popularity")

Out[26]: Text(0,0.5,'Popularity')
```




```
In [27]: popularity_genre2 = df.split.groupby(['genres']).max().popularity
         popularity_genre2
```

```
Out[27]: genres
         Action             32.985763
         Adventure          32.985763
         Animation           8.691294
         Comedy              8.691294
         Crime               9.335014
         Documentary         1.005772
         Drama              24.949134
         Family              8.691294
         Fantasy            11.173104
         Foreign             0.741302
         History             8.110711
         Horror              8.411577
         Music               4.780419
         Mystery             9.363643
         Romance             6.715966
         Science Fiction    32.985763
         TV Movie            1.844119
         Thriller           32.985763
         War                 8.110711
         Western             9.110700
         Name: popularity, dtype: float64
```

```
In [28]: # Answer: Most Popular Genre

In [29]: # Question: Which is the most popular film/ Genre?

In [30]: max_pop = df.split['popularity'].max()
         max_pop

Out[30]: 32.985762999999999

In [31]: df[df['popularity'] == max_pop]

Out[31]:    popularity     budget     revenue  original_title  runtime  \
         0   32.985763  150000000  1513528810  Jurassic World      124

                                             genres  release_year
         0  Action|Adventure|Science Fiction|Thriller          2015

In [32]: # Answer: the most popular film is Jurassic World, which contains to the 4 genres: Acti

In [33]: df.split.head()

Out[33]:    popularity     budget     revenue                original_title  runtime  \
         0   32.985763  150000000  1513528810                Jurassic World      124
         1   28.419936  150000000   378436354             Mad Max: Fury Road      120
         2   13.112507  110000000   295238201                      Insurgent      119
         3   11.173104  200000000  2068178225  Star Wars: The Force Awakens      136
         4    9.335014  190000000  1506249360                      Furious 7      137

            release_year     genres
         0          2015     Action
         1          2015     Action
         2          2015  Adventure
         3          2015     Action
         4          2015     Action

In [34]: popularity_genre3 = df.split.groupby(['genres'])
         popularity_genre3

Out[34]: <pandas.core.groupby.DataFrameGroupBy object at 0x7f76bf113828>

In [35]: #  Question: Have films become more and more popular over time?
         popularity_year = df.split.groupby(['release_year']).max().popularity.plot(kind='bar',
         popularity_year.set_xlabel("Year")
         popularity_year.set_ylabel("Count in Mio.")

Out[35]: Text(0,0.5,'Count in Mio.')
```



In [36]:
```
# Answer: the popularity do not rise over time.
```

In [37]:
```
# Disply the mean, min and max of popularity over the time (values in Mio.)
df.split.describe()
```

Out[37]:

	popularity	budget	revenue	runtime	release_year
count	26955.000000	2.695500e+04	2.695500e+04	26955.000000	26955.000000
mean	0.706112	1.750781e+07	4.744365e+07	102.800408	2000.698423
std	1.114979	3.460893e+07	1.322100e+08	30.373314	12.764378
min	0.000065	0.000000e+00	0.000000e+00	0.000000	1960.000000
25%	0.224628	0.000000e+00	0.000000e+00	90.000000	1994.000000
50%	0.411324	1.130000e+02	0.000000e+00	99.000000	2005.000000
75%	0.774737	2.000000e+07	3.053601e+07	112.000000	2011.000000
max	32.985763	4.250000e+08	2.781506e+09	900.000000	2015.000000

In [38]:
```
#  Answer: the mean of popularity over the period is only 0.706.
# The data shows a hugh span between the maximum (32.985763 Mio,) and minimum (0.000065
# In the diagram one can see, the most popular film was released in 2015. There is no r
```

In [39]:
```
df.groupby(['release_year']).groups.keys()
```

Out[39]:
```
dict_keys([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972
```

In [ ]:

In [ ]:

1.1.4 Research Question 2

What kinds of properties are associated with movies that have high investments?

In [40]:
```
df.describe()
# mean of the Budget 1.465531e+07
```

Out[40]:

	popularity	budget	revenue	runtime	release_year
count	10842.000000	1.084200e+04	1.084200e+04	10842.000000	10842.000000
mean	0.647461	1.465531e+07	3.991138e+07	102.138443	2001.314794
std	1.001032	3.093971e+07	1.171179e+08	31.294612	12.813617
min	0.000065	0.000000e+00	0.000000e+00	0.000000	1960.000000
25%	0.208210	0.000000e+00	0.000000e+00	90.000000	1995.000000
50%	0.384532	0.000000e+00	0.000000e+00	99.000000	2006.000000
75%	0.715393	1.500000e+07	2.414118e+07	111.000000	2011.000000
max	32.985763	4.250000e+08	2.781506e+09	900.000000	2015.000000

In [41]:
```
#Question: more budget = longer runtime?
```

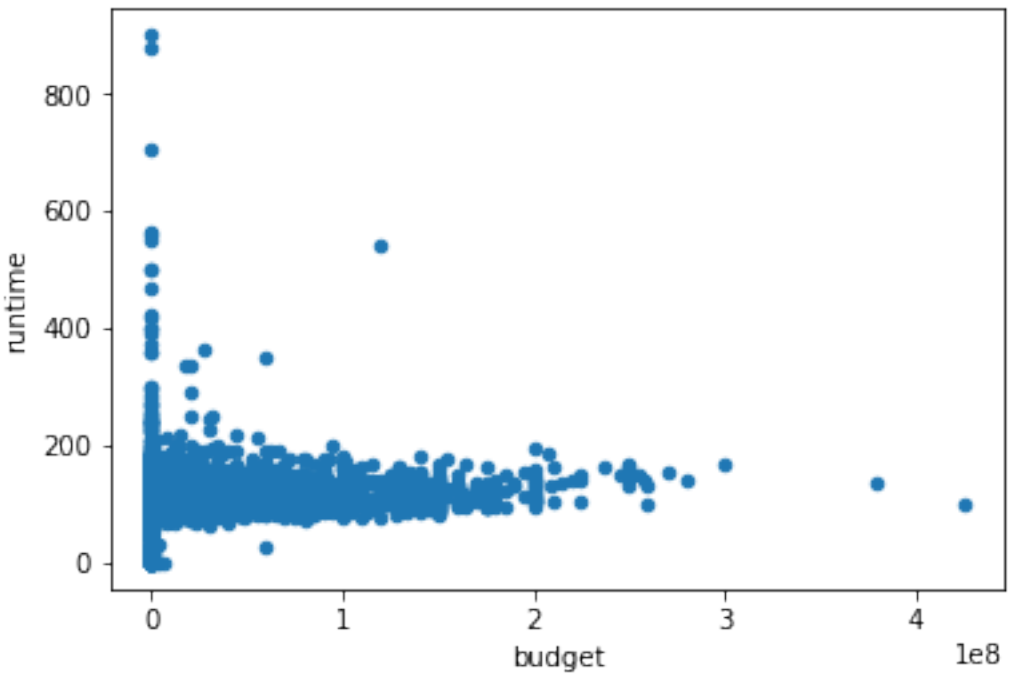
In [42]:
```
def correlation(x, y):
    std_x = (x - x.mean()) / x.std(ddof=0)
    std_y = (y - y.mean()) / y.std(ddof=0)

    return (std_x * std_y).mean()
```

In [43]:
```
correlation(df['budget'], df['runtime'])
```

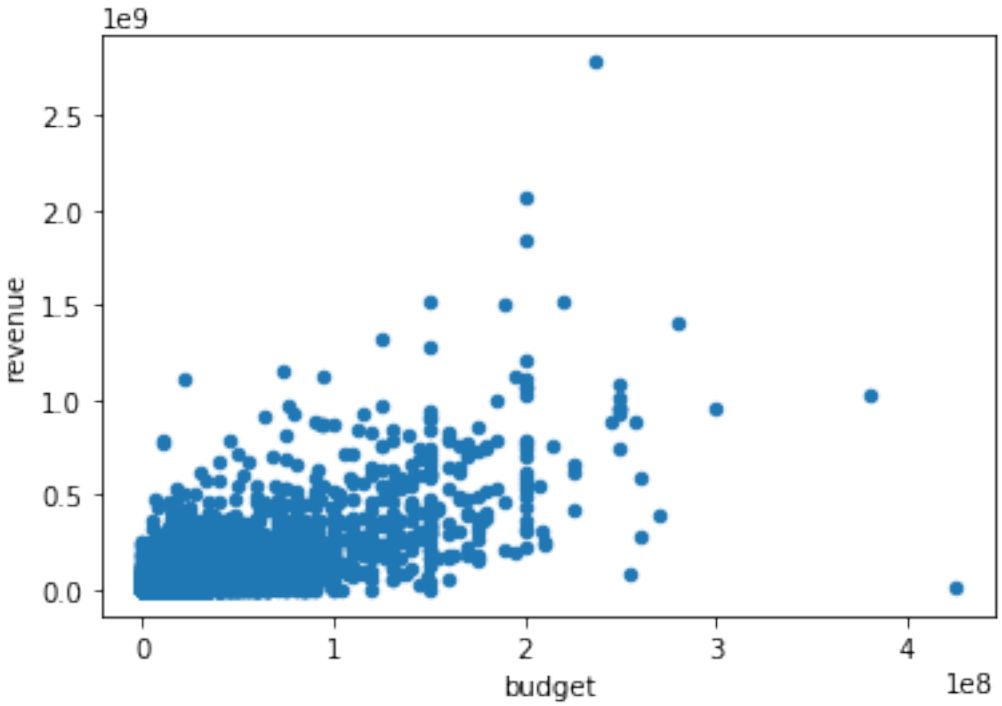
Out[43]:
```
0.19107896995964305
```

In [44]:
```
df.plot(x="budget", y="runtime", kind="scatter")
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f76bec2dd68>
```



```
In [45]: #Answer: there is a positive correlation between the budget and the runtime, but not a
         # One can say, the more budget was available, the longer the film would last.

In [46]: # Question: more budget = more revenue?

In [47]: correlation(df['budget'], df['revenue'])

Out[47]: 0.73485113100762367

In [48]: df.plot(x="budget", y="revenue", kind="scatter")

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7f76bebe46a0>
```




```
In [49]: # Answer: positive correlation
         # the more budget was available, the higher the profit.

In [ ]:
```

Conclusions

Question 1: How did the release of films change over time?

In the exploration we found out that more films were released in one year from year to year. However, the most popular films per year show no general increase in popularity.

The most used genre for films over time is drama, where films that have one or more genres have been screened.

the most popular genre or the corresponding film "Jurassic World" comes from the genre Action, Adventure, Science Fiction and Thirller.

Question 2: What kinds of properties are associated with movies that have high investments?

in the exploration we found out, that there is a position relationship between budget and runtime of a film and also betweeen budget and revenue of a film. So we can say, the more budget was available, the higher the profit and the more budget was available, the longer the film would last.

1.2 Submitting your Project

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [51]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Investigate_a_Dataset.ipynb'])
```

```
Out[51]: 0
```

```
In [ ]:
```

```
In [ ]:
```